REMARKS

The Office Action mailed December 20, 2005, and the references cited by the Examiner have been carefully reviewed by Applicants.

Claims 32-34 are pending in this case. Applicants submit that, for the reasons discussed below, the pending claims are in condition for allowance, and Applicants earnestly seek such allowance.

Review of the References

Chiang et al. (U.S. Patent No. 6,948,174, hereinafter "Chiang") discloses a method of and a system for processing an application request on an end user application and an application server. This is accomplished by initiating the application request on the end user application in a first language (such as a markup language) with a first application program (such as a Web browser), and transmitting the application request to the server and converting the application from the first language of the first end user application to a language running on the application server, processing the application request on the application server, and transmitting the response from the application server back to the end user application, and converting the response from the language running on the application server to the language of the end user application. Chiang does not discuss the use of the Java Message System, or of the difficulties of integrating a Java Message System with other systems.

Response to Rejections under Section 103

The presently disclosed system and method, as claimed, make use of a Java Message System (JMS). While JMS is released as an API (Application Programming Interface) for Java, not all Java systems use JMS. The use of JMS on the Java platform when communicating with other platforms has unique advantages and difficulties when implemented. These unique advantages and difficulties have not been addressed in the references, because the cited reference does not use JMS on a Java platform as does the present system. Applicants respectfully submit that simply because a reference uses Java does not teach or suggest receiving, interpreting, or transmitting messages sent in the JMS format.

In the Office Action dated December 20, 2005, Claims 32-34 were rejected under 35 USC § 103(a) as being unpatentable over Chiang et al. (U.S. Patent No. 6,948,174). The Office Action suggested, in the rejection of the single independent claim at issue, the following:

> *"As to claim 32, Chiang teaches a method for brokering message between middleware systems comprising: ... communicating a message from a Java based system into a Java message format (col. 10, lines 18-45 mapping the message in the Java format onto the fields in a structured event format to a middleware brokering system (col. 10, lines 18-45)"*

Applicants are unable to locate any reference to the use of a Java Message System (JMS). As will be discussed more fully below, the use of the JMS system presents challenges not addressed by the cited reference. Chiang discloses only the use of a Java application layer in the following passage:

> *"The Common Application Metamodel tool, method, and system is especially useful for providing a data transformer that is bi-directional between a client application and a server application, transmitting commands and data both ways between, for example,*

> *a Java, HTML, XML, C, or C++ application and a COBOL, PL/I, or High Level Assembler application, or, between an HTML or XML application and a Java, C, or C++ application, or between a Java application and a C or C++ application." (Col. 3, ll 42-52)*

While the cited art may have allowed a system with a Java implementation to interact with messages in a Cobol format, it does not allow for the interaction with messages within a JMS. Specific support must be in place prior to the Java system being capable of using messages in a Java format. When Chiang refers to Java, the reference is limited only to the platform on which pre-compiled computer code is executed as an application, not the format on which data of a specialized format, such as JMS, are sent and received.

As previously disclosed, JMS is released as an API (Application Programming Interface) for Java. Other API mechanisms for messaging may be used with the Java, such as the GMail API for Java. This API does not support JMS messaging, yet does use the Java Platform. The Office Action asserts that Java messaging is disclosed in the cited references, but does not indicate where JMS messaging is disclosed in the cited references. Applicants respectfully point out that simply because messaging is preformed on the Java platform does not mean that the messaging system uses JMS. For this reason, Applicants submit that the cited reference fails to teach, disclose, or suggest communication via a JMS system in a JMS message format as claimed in Applicants' claim 32. If Applicants have overlooked such a reference, Applicants respectfully request that the next Office Action specifically indicate where such a reference may be found.

None of the API mechanisms previously used with the Java platform have all of the same unique characteristics and implementation difficulties associated with JMS messaging, such as asynchronous messaging (A JMS provider can deliver messages to a client as they arrive; a client does not have to request messages in order to receive them), and enhanced reliability (the JMS API can ensure that a message is delivered once and only once.)

Therefore, JMS, as pointed out in the Applicants' original disclosure, is a special kind of messaging system that has unique properties. Referring to Paragraph [0046] of Applicants' original disclosure:

> When the JMS network **330** needs to publish a message, a JMS application places the message in the form of a JMS MapMessage and sends the MapMessage to the JMS adapter **350**. The JMS adapter **350** then converts the MapMessage into a structured event by mapping the fields in the MapMessage to the fields in the structured event. The domain name and type name properties of the message are concatenated by the JMS application in the form DomainName:TypeName and are placed in the JMSType field in the header section of the MapMessage. The JMS adapter **350** maps the DomainName portion of the concatenated JMSType field onto the domain_name **425** field in the fixed header **422** of the structured event and maps the TypeName portion of the concatenated JMSType field onto the type_name **426** field in the fixed header **422** of the structured event. If the JMS application does not properly set the JMSType field, the JMS adapter **350** populates this field with a default value. The MapMessage does not supply a value for the event_name **427** field in the structured event. This field is instead set by the message brokering server **360** upon receipt of a message.

Chiang does address the special problems associated with the JMS format. For instance, Chiang does not disclose correcting errors within a JMS by populating fields with default values when there is an improperly placed value. The cited reference does not teach or suggest the

use of a JMS or attempt to address the unique requirements of the JMS.

In addition, the method employed by the referenced art teaches away from using the present invention. The present application teaches a middleware capable of mapping the message in Cobol copybook, CORBA, and JMS formats onto the fields in a structured event format as well as communicating messages to a middleware brokering system. In contrast, the referenced art teaches away from this method of the middleware being capable of this mapping in lieu of the use of a "stub" which must be run on both the client and the server. The referenced art describes the use of the stub as follows:

> *A "stub" is a small program routine that provides static interfaces to servers. Precompiled stubs define how clients invoke corresponding services on the server. The stub substitutes for a longer program on the server, and acts as a local call or a local proxy for the server object. The stub accepts the request and then forwards it (through another program) to the remote procedure. When that procedure has completed its service, it returns the results or other status to the stub which passes it back to the program that made the request. Server services are defined in the stub using an Interface Definition Language ("IDL"). The client has an IDL stub for each server interface that it accesses and includes code to perform marshaling. Server stubs provide static interfaces to each service exported by the server.* (Col. 80, ll 51-64)

The referenced art requires an application program, executed on the client machine, and another application program executed on the server side in order for the middleware to function. This requires the client to run a messaging program as well as a conversion program, and the server to run a similar set of programs. In contrast, the currently disclosed method does not require a stub to be run on the client as all conversion is possible within the middleware itself. If

a Cobol or CORBA message is used by a client, it is seamlessly transmitted to the server without a need for translation.

The Office Action continues by asserting that the Chiang reference is capable of:

> *"communicating a message from a Java format to the structure event format to the middleware brokering system (col. 10, lines 46-61); communicating a message from CORBA system in a structured event format to the middleware brokering system (col. 10, lines 46-61); using the middleware broker to determine the destination for each of the message from the Java, CORBA, and mainframe systems (col. 10, lines 46-61); and directing each of the messages to the appropriate one of the JMS, CORBA, and mainframe systems (col. 10, lines 46-61)"*

Applicants respectfully reiterate the previously discussed arguments as to the omission of any reference to the use of a JMS. It is respectfully asserted that the cited reference must be modified in order to interact with a JMS system. It is further respectfully asserted that modifications to the prior art using the claimed invention as a model for the modifications is impermissible. Unless some teaching exists in the prior art for the suggested modification, merely asserting that such a modification would be obvious to one of ordinary skill in the art is improper and cannot be used to meet the burden of establishing a *prima facie* cases of obviousness.

Therefore, absent some teaching, suggestion, or incentive in the prior art, Chiang cannot be construed to anticipate or make obvious Applicants' disclosure. Chiang requires the additional step of the use of a "stub" and does not address the unique challenges presented by the JMS format. As a result, absent any teaching, suggestion, or incentive from the prior art to make the proposed modifications, the presently claimed invention can be reached only through the

impermissible use of hindsight with the benefit of Applicants' invention as a model.

The Office Action notes that the JMS format is not taught by the Chiang reference and attempts to cure this by asserting that it is obvious to use JMS at the time of the invention.

> *Chiang does not explicitly teach the use of JMS messages with Java. Chiang teaches the use of numerous software applications and the use of Java, so though not explicitly mentioned Java Message Service format was a well-known way to communicate with applications at the time of the invention. It would have been obvious to one of ordinary skill in the Computer Networking art at the time of the invention to combine the teachings of Chiang regarding the exchange of data between numerous applications on varying platforms with the use of JMS messages because the various applications discussed by Chiang could include applications that use JMS messages, without departing from the scope of the Chiang invention.*

Applicants respectfully reiterate the previous arguments as to the unique nature of a JMS, and the requirement of the "stub" by the Chiang reference. The present innovations recognize the identified problems with the transmission of JMS messages and have found innovative ways to address them. Chiang does not teach the problem of transmitting data within a JMS or its source. Instead, Chiang is directed towards only translating messages with precompiled stubs.

The Office Action also rejected claims 33 and 34 under 35 USC § 103(a) as being unpatentable over Chiang. Dependent Claim 33 and Claim 34 depend directly from independent Claim 32 and incorporate all the limitations thereof. Thus, Applicants respectfully request withdrawal of this rejection and full allowance of all claims.

Conclusion

Applicants respectfully submit that the application in its present form is in condition for allowance. If the Examiner has any questions or comments or otherwise feels it would be helpful in expediting the application, Examiner is encouraged to telephone the undersigned at (972) 731-2288. Applicants intend this communication to be a complete response to the Office Action mailed on December 20, 2005.

The Commissioner is hereby authorized to charge payment of any further fees associated with any of the foregoing papers submitted herewith, or to credit any overpayment thereof, to Deposit Account No. 21-0765, Sprint.

Respectfully submitted,

Date: 3/14/2006

Michael W. Piper
Reg. No. 39,800

ATTORNEY FOR APPLICANTS

CONLEY ROSE, P.C.
5700 Granite Parkway, Suite 330
Plano, Texas 75024
(972) 731-2288
(972) 731-2289 Facsimile